(A tesztelés Postman- ben történt)

|  |  |  |
| --- | --- | --- |
| OpenAPI elemek megvalósítása | Pontszám | Megvalósítás helye: |
| * **Szabványos swagger raw endpoint** | 1 | swagger.yaml 147.sor |
| * **JSON támogatás** | 1 | swagger.yaml  sorok(16,19)  authentication.js (27.sor),data.js(17.sor),invoice.js(25.sor) register.js(20.sor) |
| * Swagger request és response validation használata | 1 | swagger.yaml sorok(51,56,60,75,80,84,88,106,110,114,137,141, 145,170,174,178,206,210,214,242,246,250) |
| * **GET, PUT, POST, DELETE végpontok** | 4 | GET: swagger.yaml 91.sor, 117.sor POST: swagger.yaml 39.sor, 63.sor PUT: swagger.yaml 181.sor, 217.sor DELETE: swagger.yaml 150.sor |
| * Újrahasznosítható objektum definíciók használata minden végponthoz | 2 | swagger.yaml sorok(51,56,60,75,80,84,88,106,110,114,137,141, 145,170,174,178,206,210,214,242,246,250) |
| * Required mezők használata az objektum definíciókban és paraméterekben | 2 | swagger.yaml sorok(49,73,100,126,131,159,164,190,195, 200,226,231,236,254,260,275,282,293,300,315,330) |
| * Query string paraméter használat lekérdezésnél | 1 | swagger.yaml 198.sor,234.sor |
| * Hibakezelés: Közös hiba definíció használata, minden endpoint által használva, controllerekben megvalósítva | 3 | swagger.yaml sorok(51,56,60,75,80,84,88,106,110,114,137,141, 145,170,174,178,206,210,214,242,246,250) def sor(252- től) |
| * Hibakezelés: HTTP hibakódok használata különböző hiba esetekre (pl. hiányzó objektum, authentikációs hiba, hiányzó jogosultság, egyéb szerver hiba), controllerekben megvalósítva | 3 | swagger.yaml sorok(52,76,102,133,166,202,238)  401 : authentication.js,data.js,invoice.js 400: data.js,invoice.js,register.js |
| * Tag-ek használata végpontok csoportosítására | 1 | swagger.yaml sorok(27,43,67,94,120,153,184,220) |
| * XML támogatás minimum 1 végpontra | 2 |  |
| API authentikáció - Swagger Security |  |  |
| * **Globális session alapú authentikáció minden endpointra (API Key)** | 5 | swagger.yaml sorok(21,26) |
| * **Login és Signup végpontok, authentikációs kivételekkel** | 2 | authentication.js register.js |
| API üzleti logika - Controllers |  |  |
| * Adattárolás (in-memory vagy perzisztens) használata | 2 | db.js (in-memory) |
| * **A választott téma szerinti logika megvalósítása (számítás, nem csak CRUD)** | 6 | invoice.js (function pay(…) ->104.sor -> ÁFA (VAT) számítás) |
| API Gateway használat - Kong |  |  |
| * **API Gateway használata reverse proxy-ként** | 3 | docker-compose.yaml |
| * API Key használata kliens azonosításra (mobil, web) | 4 | kong\_config.txt 35.sor |
| * Rate limit használata (globális) | 3 | kong\_config.txt 38.sor |
| * Rate limit használata (klienstől függő - Kong consumer) | 4 | kong\_config.txt 43.sor |
| * Dinamikus terhelés (load balancing) | 4 | kong\_config.txt 6.sor, 11.sor |
| Docker Compose |  |  |
| * **Működő Docker-compose deployment Kong-gal és OpenAPI-val (verzio min 3)** | 5 | docker-compose.yaml |
| * Verziózott Docker image-ek használata | 3 | docker-compose.yaml 55.sor, 61.sor |
| * Health check definiálása az API-hoz | 2 | docker-compose.yaml 49.sor |